DOI: 10.3745/KIPSTD.2010.17D.5.327

기능점수 기반 소프트웨어 공식

이 상 운†

요 약

본 논문은 기능점수 소프트웨어 규모에 기반하여 개발노력과 일정과의 관계를 유도하는 소프트웨어 공식을 제안하였다. 기존의 소프트웨어 공식은 라인수에 기반을 두고 있다. 라인수는 개발언어에 따라 큰 차이를 보여 소프트웨어 규모 추정에 어려움이 많이 지적되고 있다. 먼저 라인수를 기능점수로 변환하는 방법을 고려하였다. 그러나 이 방법은 개발언어별로 라인수와 기능점수간 변환비율이 명확히 결정되지 않고 있고, 또한 특정 개발언어에 대해서는 변환비율이 제시되어 있지 않아 소프트웨어 공식을 유도하는데 실패하였다. 따라서 기능점수에 기반하여 개발된 대용량의 프로젝트 데이터를 대상으로 소프트웨어 공식을 직접 유도하였다. 첫 번째로 개발 프로젝트들 중에서 타당한 개발기간이 설정된 데이터들을 분류하였다. 두 번째로, 이 데이터에 대해 회귀분석을 통해 기능점수와 개발노력, 기능점수와 개발기간과의 관계를 유도하였다. 마지막으로 이들 관계로부터 소프트웨어 공식을 유도하였다. 제안된 모델은 라인수 기반의 모델이 갖고 있는 적용상 문제점들을 해결하여 실무에 쉽게 적용이 가능한 장점을 갖고 있다.

키워드: 불가능 영역, 실제 영역, 최적 영역, 비현실적 영역, 비용-일정 대안분석

Software Equation Based on Function Points

Sang-Un Lee†

ABSTRACT

This paper proposed software equation that is relation with effort and duration based on function point (FP) software size. Existent software equation based on lines of code (LOC). LOC sees big difference according to development language and there are a lot of difficulties in software size estimation. First, considered method that change LOC to FP. But, this method is not decided definitely conversion ratio between LOC and FP by development language. Also, failed though the conversion ratio motives software formula because was not presented about specification development language. Therefore, we derived software formula directly to large project data that was developed by FP. Firstly, datas that reasonable development period is set among development projects. Secondly, FP through regression analysis about this data and effort, motived relation with FP and duration. Finally, software equation was derived from these relation. Proposed model solves application problems that LOC-based model has and has advantage that application is possible easily in business.

Keywords: Impossible Region, Practical Region, Optimal Region, Impractical Region, Cost-Schedule Trade-off

1. 서 론

소프트웨어 개발의 성공 여부는 고객이 요구하는 품질 수준 (Quality)을 만족하는 제품을 주어진 일정 (On Schedule or Duration)과 예산범위 내 (On Cost or Budget)에서 납품할 수 있느냐로 결정된다[1]. 소프트웨어 개발에 소요되는 노력 (Effort, E)과 기간 (Duration, T)은 소프트웨어 제품의 크기 (규모, Size)로부터 유도된다. 소프트웨어 규모와 노력의 양이 추정되었을 경우, 어느 정도의 개발기간이 소요될 것인가 또는 개발인원을 추가로 투입하면 얼마나 기간을 단축시킬 수 있는가는 프로젝트 계획을 작성하는 관리자에게 필수적으로 요구되는 정보이다. 이러한 3개 척도들 (Metrics) 간의 관계를 표현한 것이 Putnam과 Myers가 제안한 소프트웨어 공식 (Software Equation, *SE*)이다[2-6]. 소프트웨어 공식은 소프트웨어 규모 (크기)로 라인 수 (Source Lines Of Code, *SLOC*)를 채택하고 있다. 그러나 라인 수를 계산하는 방식에도 여러 가지 문제점을 갖고 있다[7-10]. 이러한 문제점으로 인해 소프트웨어 척도 분야에서는 기능점수 (Function Point, *FP*), 완전 기능점수 (Full Function Point, *FFP*), 유스케이스 점수 (Use-Case Point, *UCP*) 또는 특징 점수 (Feature Point) 등 다양한 기법들이 개발되었다[11-16]. 현재는 *FP*가 가장 일반적으로 적용되고 있으며, *FP*의 문제점을 보완한 *FFP*로 전환되고 있는 실정이다.

† 정 회 원 : 강릉원주대학교 멀티미디어공학과 부교수
논문접수 : 2010년 3월 10일
심사완료 : 2010년 4월 6일

기능점수 규모에 기반한 소프트웨어를 개발한다고 가정하자. 이 경우 개발노력, 개발기간과 더불어 추정된 개발노력과 개발기간과의 관계로부터 비용-일정 타협을 수행하기 위해서는 기존의 *SLOC* 기반의 소프트웨어 공식을 적용해야만 한다. 그러나 *SLOC*를 *FP*로 변환하는데 따른 문제점과 더불어 기존의 소프트웨어 공식이 갖고 있는 근본적인 문제점들로 인해 적절한 소프트웨어 공식을 유도하는데 실패하여 프로젝트 계획에 활용하지 못하고 있다.

본 논문은 기존의 *SLOC* 기반의 소프트웨어 공식이 갖고 있는 문제점들을 해결하여 *FP* 기반의 소프트웨어 공식을 제안한다. 본 제안된 소프트웨어 공식을 적용함으로서 보다 현실에 적합한 프로젝트 관리를 수행할 수 있을 것이다. 2장에서는 *SLOC* 기반 소프트웨어 공식과 문제점, *FP* 기반의 소프트웨어 공식 개발시 제기되는 문제점을 살펴본다. 3장에서는 데이터 표본 추출 방법과 소프트웨어 공식 유도 과정을 통해 *FP* 기반의 소프트웨어 공식을 제안한다. 4장에서는 제안된 소프트웨어 공식을 분석하고 평가해 본다.

2. 관련 연구와 문제점

2.1 *SLOC* 기반 소프트웨어 공식과 문제점

Putnam과 Myers[3-6]는 "어떤 생산성 수준에 도달한 개발자는 일정 기간 동안 노력을 투입하면 일정한 신뢰 수준의 산출물을 얻는다."라는 법칙에 근거하여 식 (1)의 관계를 도출하였다.

$$Work\ Product(at\ a\ Reliability\ level) =$$

$$Effort—a\ Time\ interval\ at\ a\ Productivity\ level.$$

$$Size(at\ Defect\ rate) = Effort \cdot Time \cdot Process\ Productivit.$$

$$Size = E^a \cdot T^b \cdot PP \tag{1}$$

여기서, *Size*는 SLOC, *E*는 년 인원 (Person-Years), *T*는 년 (Years) 단위를 적용하고 있다. 프로젝트의 일정은 순차적인 작업의 제약사항에 영향을 받으며, 임의의 시점에서 최대로 투입할 수 있는 노력 (인원 수)은 독립적인 하위 작업들의 수에 의존하기 때문에 개발일정도 생산성에 영향을 미친다. 그러나 전통적인 생산성은 개발일정을 고려하지 않고 *SLOC/E*로 계산되어 소프트웨어 개발 생산성 척도로는 적절하지 않다[17]. 따라서 개발조직의 생산성이 노력과 일정 모두에 영향을 받도록 고려한 것이 *PP*이다. *PP*는 어느 한 프로그래머의 생산성이 아닌 개발조직 전체의 생산성이다. 이 용어는 처음에는 상수로 기술 제약사항 (Technology Constraint)으로, 다음에는 전체 조직에 적용하기 위해 조직의 생산성 (Organizational Productivity)으로 변경되었으며, 최근 들어서는 "공정" 용어의 대중성으로 공정 생산성 (Process Productivity)으로 사용되고 있다[5].

모수 a와 b는 과거 개발된 소프트웨어 프로젝트들을 대상으로 $SLOC = xE^{0.33}$와 $SLOC = yT^{1.33}$을 유도하여 식 (2)의 소프트웨어 공식 (SE, Software Equation)을 제안하였다[3, 18].

$$SLOC = PP \cdot (\frac{E}{B})^{\frac{1}{3}} \cdot T^{\frac{4}{3}} \tag{2}$$

여기서, *B*는 숙련도 인자 (Special Skill Factor)로 규모의존 모수라고도 한다. 이 모수는 소프트웨어 규모가 작을 경우 노력의 효율성이 보다 좋도록 하기 위해 작은 규모의 소프트웨어에 보다 큰 가중치를 부여하는 효과를 나타내며 <표 1>에 제시되어 있다[19, 20].

SLOC, B, E, T, a, b 들을 대입하면 $PP = \frac{SLOC}{(E/B)^{1/3} \cdot T^{4/3}}$ 식으로 계산된다. 그러나 *PP* 값은 <표 2>에서 알 수 있듯이 754에서 3,524,578의 범위를 갖고 있어 *PP* 값으로는 개발조직의 생산성 수준이 어느 정도인지 판단이 불가하여 생산성 지표 (Productivity Index, *PI*)를 사용하고 있으며, $1 \le PI \le 40$의 범위를 갖고 있다. Putnam과 Myers가 제시한 PP와 PI는 <표 2>에 제시하였다[2, 21].

식 (2)로부터 노력 (비용)과 일정을 타협할 수 있는 식 (3)이 유도될 수 있다.

$$E \cdot T^4 = B(\frac{Size}{PP})^3 = K\ (Constant) \tag{3}$$

식 (3)에 따라 개발조직의 생산성 (*PP*)과 개발할 소프트

〈표 1〉 *B* 값 적용 기준

Size(KLOC)	5-15K	20K	30K	40K	50K	>70K
B	0.16	0.18	0.28	0.34	0.37	0.39

〈표 2〉 *PP*와 *PI* 값 변환 기준

PI	PP	시스템명	PI	PP	시스템명
1	754	-	21	92,736	-
2	987	마이크로코드	22	121,393	-
3	1,220	-	23	150,050	-
4	1,597	펌웨어 (ROM)	24	196,418	-
5	1,974	실시간 내장형, 항공전자	25	242,786	-
6	2,584	-	26	317,811	-
7	3,194	레이더 시스템	27	392,836	-
8	4,181	명령 & 통제	28	514,229	-
9	5,186	공정관리	29	635,622	-
10	6,765	-	30	832,040	-
11	8,362	통신	31	1,028,458	-
12	10,946	-	32	1,346,269	-
13	13,530	시스템소프트웨어, 과학시스템	33	1,664,080	-
14	17,711	-	34	2,178,309	-
15	21,892	-	35	2,692,538	-
16	28,657	비즈니스 시스템	36	3,524,578	-
17	35,422	-	37	-	-
18	46,368	-	38	-	-
19	57,314	-	39	-	-
20	75,025	-	40	-	-

웨어의 규모 (SLOC)만 알고 있다면 개발에 투입할 노력 (E)과 일정 (T)간에 타협이 가능하다. 또한, 인력을 점진적으로 증가시키는 곡선의 기울기 MBP (Manpower Buildup Parameter) $= K/T^3$와 MBI (Manpower Buildup Index) $= E/T^3$를 적용하면 개발인력의 프로파일을 구할 수 있다.

소프트웨어 공식에서 소프트웨어 규모로 $SLOC$를 사용 시 발생되는 문제점은 다음과 같다[7].

- $SLOC$에 대한 일반적으로 받아들일 수 있는 정확한 정의가 부족하다. Jones[8]는 라인을 계산하는 방법에 대해 11개의 변형된 방법을 확인했으며, 라인 계산 방법에 따라 약 500%의 불확실성을 가지고 있음을 지적했다.
- $SLOC$는 언어에 종속되어 있다[9]. 다른 언어를 사용하여 개발된 프로젝트들에 대해 상호간에 직접 비교하기가 불가능하다. 예를 들면, 고차원 언어의 라인 당 투입되는 시간은 저차원 언어보다 많다. 또한 동일한 기능을 제공하기 위해 고차원 언어에 대해 보다 작은 SLOC가 요구됨은 자명한 사실이다.
- 요구분석 또는 설계단계에서 정확한 $SLOC$ 추정이 어려우며, 코딩이 종료된 후 측정이 가능하다[10]. 그러나 소프트웨어 개발노력은 프로젝트 착수 초기에 보다 정확히 추정하여 사업관리 측면이나 자원 재분배 측면에서 활용할 수 있어야만 한다.
- 특정한 하나의 관점인 길이만을 고려하고 있으며, 소프트웨어의 기능성 또는 복잡도를 고려하지 못해 소프트웨어 특성을 정확히 반영하지 못한다.

$SLOC$를 이용한 개발노력 추정 모델의 단점을 보완하고자 소프트웨어의 복잡도와 기능성에 대한 광범위한 연구가 이루어졌으며, 시스템의 기능성으로 소프트웨어의 규모나 개발노력을 추정하는 FP (Function Point)기법이 개발되었다.

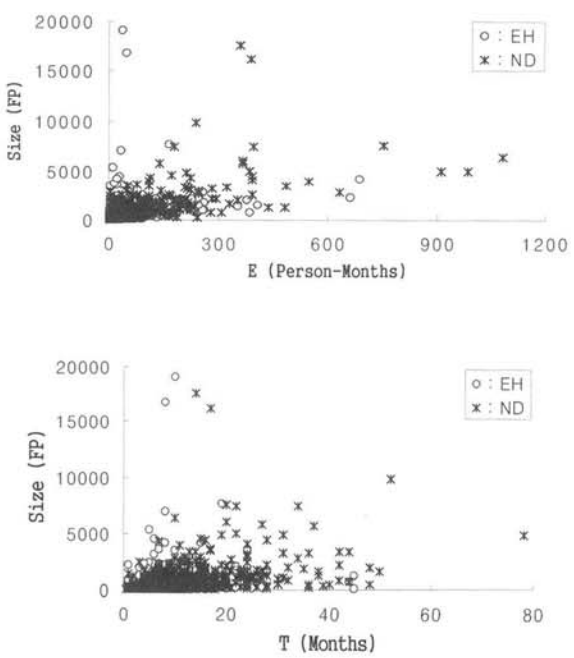
소프트웨어 규모로 $SLOC$ 적용 유무에 상관없이 기존의 소프트웨어공식은 다음과 같은 문제점을 갖고 있다. 첫 번째로, B 값의 해석 및 적용 문제이다. <표 1>에서 주어진 규모를 범위의 최대 값으로 해석 하였을 경우, 5~15K = 0.16, 15.01~20K = 0.18, 20.01~30K = 0.28, 30.01~40K = 0.34, 40.01~50K = 0.37, 70K 이상 = 0.39로 적용할 수 있다. 여기서, 0.01~5K, 50.01~60K, 60.01~70K 범위에 속하는 소프트웨어는 어떤 값을 적용해야 하는지가 문제로 제기된다. 또한, 70K~수백 K에 속하는 광범위한 규모의 소프트웨어가 모두 동일한 0.39로 적용해도 문제가 없는지에 대한 타당한 근거가 제시되어 있지 않다. 두 번째로, $SLOC$는 코딩 단계에서 소요되는 노력에만 관련되어 있다. 코딩 단계는 소프트웨어 개발 총 노력의 약 10~20% 밖에 차지하지 못하며, 이 노력이 개발 전 과정에 투입되는 노력을 대표할 수 없다. 세 번째로, 소프트웨어 공식은 1070~80년대의 폭포수 모델을 적용한 대형 프로젝트로 장기간 개발되는 형태로 년 인원과 년을 단위로 사용하고 있다. 그러나 최근의 소프트웨어 개발은 UP, Agile 방법론으로 단기간 (1~2년)에 개발을 완료하는 추세이다. 따라서 년 인원과 년 단위를 적용하는데 어려움이 있다. 마지막으로, 소프트웨어 개발시 적용되는 모든 공식의 근간이 되는 것은 $Size = xE^{0.33}$와 $Size = yT^{1.33}$의 관계이다. 이들 관계가 성립되지 않을 경우 식 (2)로부터 유도되는 모든 식들은 효용성이 없어진다. 따라서 $Size = PP \cdot E^a \cdot T^b = PP \cdot (E/B)^{0.33} \cdot T^{1.33}$ 에서, a와 b 값이 적절한 값인지 판단할 필요성이 제기된다.

2.2 FP 기반의 소프트웨어 공식 개발의 문제점

FP 기반의 소프트웨어 공식을 개발하기 위해 부딪히는 문제점은 다음과 같다. 첫 번째로, $SLOC$를 FP로 변환시켜야만 한다. 그러나 <표 3>과 같이 $SLOC$를 FP로 변환하는 표준화된 테이블이 존재하지 않으며, <표 4>와 같이 특정 개발언어에 대해서는 변환 법칙이 존재하지 않는 실정이다. 따라서 FP 기반의 소프트웨어 공식 유도가 불가능하다. <표 3>과 <표 4>의 개발언어별 프로젝트 수 분석에는 ISBSG Benchmark Release 8[22]의 2,027개 프로젝트들 중에서 개발언어가 명시된 프로젝트들을 대상으로 하였다.

두 번째로, 소프트웨어 규모로 FP를 적용하였을 경우에도 $FP \propto E^{0.33}$과 $FP \propto T^{1.33}$ 관계가 성립하는지 여부이다. 이를 검증하기 위한 실험에는 ISBSG Benchmark Release 8[22]의 2,027개 소프트웨어 프로젝트들 중에서 개발노력과 개발기간이 기술된 1,595건을 대상으로 하였으며, 이들 데이터는 성능개량 875건과 신규개발 720건으로 분류될 수 있다.

1,595건에 대한 개발노력 (E, Person-Months) vs. 소프트웨어 규모 (FP)와 개발기간 (T) vs. 소프트웨어 규모 (FP) 관계는 (그림 1)과 같으며, 회귀분석 결과 얻은 결과



(그림 1) 소프트웨어 규모와 개발노력, 개발기간 관계

〈표 3〉 SLOC/FP 변환 비율 제시 개발언어

개발언어 (개발 프로젝트 수[22])	SLOC/FP 비율					
	CS10 [23]	QSM [24]	David [25]	Jones [27]	Mcgilbon [28]	Sifri [29]
4GL(44)	20	-	-	-	64	40
Access/Access 8.0(43)	-	35	-	-	-	-
Ada/Ada 95(3)	70	154	-	-	71	71
AI Shell(0)	-	-	-	-	49	-
Algol(0)	-	-	-	-	-	106
APL(0)	-	-	-	-	32	32
APS(0)	-	86	-	-	-	-
ASP/ASP 2.0(7)	-	69	-	-	-	-
Assembler(11)	300	337	-	-	320	213
Assembler(Micro)(0)	-	-	400	-	213	-
Basic(0)	-	-	575	-	64	64
Basic-compiled(9)	-	-	-	-	91	-
Basic-Interpreted(0)	-	-	-	-	128	-
C(148)	-	162	225	128	128	-
C++(114)	-	66	80	53	29	26
Clipper(9)	-	38	60	-	-	-
COBOL(ANSI 85)(319)	100	77	175	-	91	106
Cool-Gen/IEF(29)	15	38	-	-	-	-
Culprit(0)	-	51	-	-	-	-
DBASE III(0)	-	-	60	-	-	-
DBASE IV(0)	-	52	55	-	-	-
Easytrieve+(9)	-	33	-	-	-	-
Excel(0)	-	47	-	-	-	-
Focus(3)	-	43	60	-	-	-
Forth(0)	-	-	-	-	-	64
Fortran 77(5)	100	-	210	-	105	106
FoxPro(0)	-	32	-	-	-	-
HTML(4)	-	47	-	-	-	-
Ideal(6)	-	66	-	-	-	-
Informix(0)	-	42	-	-	-	-
Java(31)	-	62	80	53	-	-
Java Script(7)	-	58	50	-	-	-
JCL(0)	-	91	400	-	-	-
Jovial(0)	-	-	-	-	105	106
JSP(0)	-	59	-	-	-	-
Lisp(1)	-	-	-	-	64	64
Logo(0)	-	-	-	-	53	-
Lotus Notes(5)	-	21	-	-	-	-
Mantis(1)	-	71	-	-	-	-
Mapper(0)	-	118	-	-	-	-
Modula 2(0)	-	-	-	-	80	-
Natural(81)	-	60	100	-	-	-
OOP(3)	30	-	-	-	-	-
Oracle /Oracle V7(100)	-	30	60	-	-	-
Oracle 2000(4)	-	41	-	-	-	-
Oracle Forms(3)	-	42	-	-	-	-
Pascal(8)	90	-	-	-	91	91
Pascal Object(0)	-	-	-	-	29	-
PeopleSoft(0)	-	33	-	-	-	-
Perl(0)	-	60	50	-	-	-
PL/I(78)	-	78	126	-	-	80
Power Builder(29)	-	32	-	-	-	-
Prolog(0)	-	-	-	-	-	64

〈표 4〉 SLOC/FP 변환 비율 미 제시 개발언어

개발언어 (개발 프로젝트 수[22])	SLOC/FP 비율					
	CS10 [23]	QSM [24]	David [25]	Jones [26]	Mcgilbon [27]	Sifri [28]
3GL(6)	-	-	-	-	-	-
5GL(1)	-	-	-	-	-	-
ABAP(4)	-	-	-	-	-	-
ABF(1)	-	-	-	-	-	-
ADS(7)	-	-	-	-	-	-
ApG(19)	-	-	-	-	-	-
CICS(5)	-	-	-	-	-	-
CLIST(1)	-	-	-	-	-	-
COBOL II(123)	-	-	-	-	-	-
COBOL MVS(12)	-	-	-	-	-	-
Col/Composer(1)	-	-	-	-	-	-
Coldfusion(3)	-	-	-	-	-	-
CSP(10)	-	-	-	-	-	-
Delphi(4)	-	-	-	-	-	-
Drift(2)	-	-	-	-	-	-
HPS(10)	-	-	-	-	-	-
IEF(1)	-	-	-	-	-	-
Ingress(1)	-	-	-	-	-	-
Jam(1)	-	-	-	-	-	-
Lex(1)	-	-	-	-	-	-
Notes Script(3)	-	-	-	-	-	-
Oracle SQL Forms(2)	-	-	-	-	-	-
Pega Workflows(1)	-	-	-	-	-	-
Periphonics(6)	-	-	-	-	-	-
Periproducer(2)	-	-	-	-	-	-
PI-Open(1)	-	-	-	-	-	-
PL/SQL(11)	-	-	-	-	-	-
Pro *C(5)	-	-	-	-	-	-
Pro *COBOL(1)	-	-	-	-	-	-

〈표 5〉 *FP vs. E, FP vs. T* 관계 분석

구분		*FP vs. E* 관계	*FP vs. T* 관계
성능개량 프로젝트	모델	$FP_{EH} = 61.9368E^{0.5099}$	$FP_{EH} = 86.9288T^{0.5323}$
	상관계수	0.3091 (30.91%)	0.1731 (17.31%)
	결정계수	0.3250(32.50%0	0.1192(11.92%)
신규개발 프로젝트	모델	$FP_{ND} = 73.7011E^{0.5538}$	$FP_{ND} = 70.6777T^{0.8107}$
	상관계수	0.6555 (65.55%)	0.4166 (41.66%)
	결정계수	0.5182(51.82%)	0.3365(33.65%)

는 <표 5>에 제시하였다. 여기서, ND는 신규개발 (New Development) 프로젝트를, EH는 성능개량 (Enhancement) 프로젝트를 의미한다. (그림 1)과 <표 5>에서 보는 바와 같이, 소프트웨어 규모와 개발노력, 규모와 개발기간 관계가 특정한 경향을 보이지 않고 산포되어 있어 거의 상호관계가 없어 보인다. 따라서 이 데이터를 이용한 관계로부터 모델을 유도하여도 효용성이 없으며, 새로운 방법을 연구할 필요가 있다.

3. 기능점수에 기반한 소프트웨어 공식

3.1 데이터 표본 추출 방법

균질(Homogeneous)의 데이터 표본을 추출하여 분석하여 보면 유익한 소프트웨어 공식을 얻을 수 있을 것이다. 통계학에서는 모집단에서 균질의 표본을 추출하기 위해 상호 배타적 서브 표본들인 층들 (Strata)로 분할하는 방법인 층화 샘플링 (Stratified Sampling) 방법을 적용하고 있다[29]. 본 논문에서는 이 방법을 적용하여 균질한 표본을 추출하기 위한 기준으로 하나의 소프트웨어를 개발하기 위해 적절한 개발기간을 설정하였는가에 초점을 맞추고자 한다. 개발기간과 개발노력 관계는 (그림 2)와 같이 여러 영역으로 분류될 수 있다.

- T_c : 최소 개발완료 시점(더 이상 단축할 수 없는 개발기간)
- T_o : 최적의 개발완료 시점(개발노력과 개발기간 고려)
- T_e : 최적의 개발완료 시점(최소의 개발비용 소요시점)
- T_n : 명목상 개발완료 시점(대부분의 프로젝트에서 평균적으로 소요되는 개발기간)
- T_s : 최대 개발완료시점(더 이상 연장시킬 수 없는 개발기간)

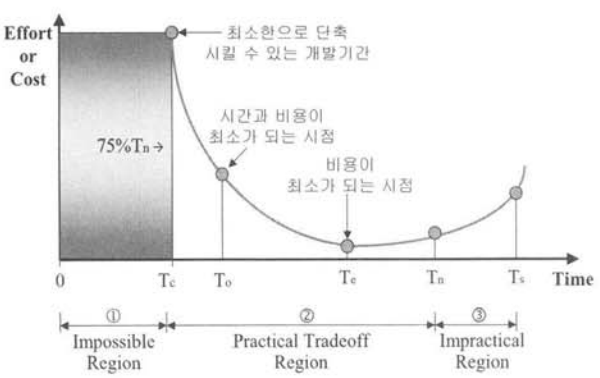
소프트웨어 개발에서는 개발인력을 무한히 추가하더라도 어느 시점 이하에서는 시스템을 성공적으로 완료할 수 없는 최소한의 개발기간이 존재하며, 이를 불가능 영역(Impossible Region) 이라 부른다[11, 28, 30, 31]. 프로젝트를 성공적으로 완료하기 위해서는 불가능영역을 회피할 수 있어야만 한다. 고객은 항상 보다 저렴하게(Cheaper), 보다 빠르게(Faster), 보다 좋은 품질(Better)의 시스템을 획득하려는 경향이 있다. 그러나, 개발자는 최소한의 개발기간으로 개발이 진행되는데 만족하지 않을 것이다. 왜냐하면 단기간에 개발할 경우 소요되는 노력이 기하급수적으로 증가하며, 이로 인해 간접비용 증가와 더불어 결함도 추가로 증하기 때문이다. 반면에 개발기간을 연장시키면 노력과 결점을 동시에 줄일 수 있다. 따라서 개발노력과 개발기간 모두를 고려하여 적절한(최적의) 값을 설정하는 것이 중요하며 이를 비용-일정 타협 (Cost-Schedule Tradeoff)이라 한다.

명목상 (Nominal) 소프트웨어의 개발 일정 (T_n)을 최대로 단축시킬 수 있는 시점 (T_c)에 대해 대부분의 연구자들은 약 75%에 한계가 있다고 동의하고 있다. 즉, $T_c = 75\% T_n$가 된다. T_c 시점에서는 T_n 시점에서 소요되는 노력의 43%가 추가로 소요된다[32]. Putnam과 Myers[1]는 일반적으로 T_s는 T_n의 약 130% 근방에 존재한다고 제시하였다. T_n 시점 이후에는 노력은 보다 감소하고 결점은 보다 작아진다. 경험에 의해 일반적으로 $T_c = \sqrt{E} (= E^{0.50})$ (Square Root Law)을, $T_n = a \cdot \sqrt[3]{E} (= aE^{0.33})$ (Cube Root Law)를 적용하고 있다[33].

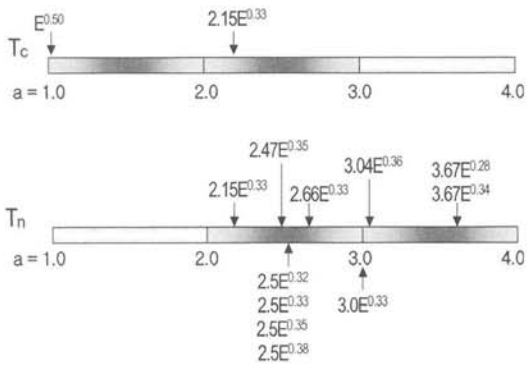
(그림 3)에서 알 수 있듯이 T_c에 대한 연구 결과를 살펴보면, Boehm[34], Putnam과 Myers[2]는 $2.15E^{0.33}$을, Carbno [31]와 Ward[35]는 $\sqrt{E} (= E^{0.50})$을 제시하였다. T_n에 대한 연구결과로 COCOMO[11]는 Organic (Basic) Mode는 $2.5E^{0.38}$, Semi-detached Mode는 $2.5E^{0.35}$, Embedded (System, Utility) Mode는 $2.5E^{0.32}$를 제시하고 있다. Pressman, Jones, Basilli와 Putnam[36]은 $2.66E^{0.33}$, Richard[37]는 $3.0E^{0.33}$ $(2.0 \le a \le 4.0)$을, Holtsinger[38]는 $3.0E^{0.33}$ $(2.5 \le a \le 4.0)$을 COCOMO II[39]는 $3.67E^{f}$ $(0.28 \le f \le 0.34)$를, Jones [26]는 $2.5E^{0.33}$을, Marin[40]은 $3.0E^{0.33}$을, Walston-Felix[41]는 $2.47E^{0.35}$를 제시하고 있다. 또한, Boyston은 $2.15E^{0.33}$을, Nelson은 $3.04E^{0.36}$을 제시하였다[42]. 이와 같이 T_c와 T_n에 대해 다양한 모델이 제시됨에도 불구하고 개발될 소프트웨어에 대해 명확히 특정한 하나의 모델을 적용할 수 있는 기준이 설정되어 있지 않다.

ISBSG Benchmark Release 8[22] 데이터들의 개발노력은 $0.03PM \le E \le 1,100PM$ 범위를 갖고 있다. (그림 2)에 제시된 개발일정 추정 모델들을 이 값에 대입하여 본 결과는 (그림 4)와 같다.

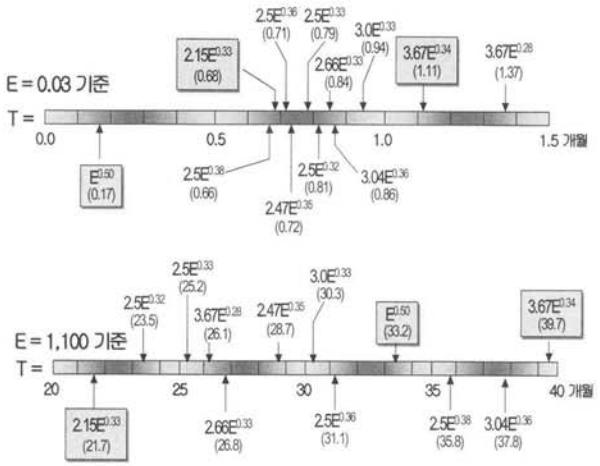
(그림 4)로부터 $T_c = 2.15E^{0.33}$은 최소값을 유지하고 있어 일단은 T_c로서 채택을 고려할 수 있다. 그러나



(그림 2) 전형적인 프로젝트의 노력과 개발기간 관계



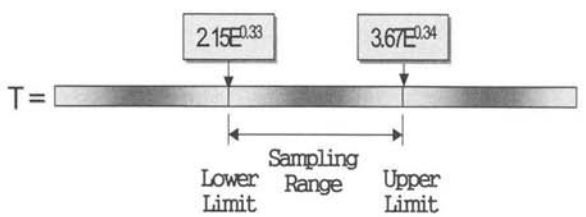
(그림 3) a 값 기준 개발일정 추정 모델 분포



(그림 4) 개발일정 모델 값 분포

$T_c = E^{0.5}$는 최소값을 가지지 못하여 T_c로서 채택이 불가함을 알 수 있다. 또한, 최대 값은 $3.67E^{0.34}$가 타당해 보인다. 결국 (그림 5)의 범위에 속하는 프로젝트들의 개발기간이 타당하게 설정되었으며, 이 범위 밖의 데이터들은 비정상적으로 개발이 진행되었다고 볼 수 있다. 따라서 표본으로 추출하기 위한 데이터 범위는 (그림 5)와 같이 설정하였다.

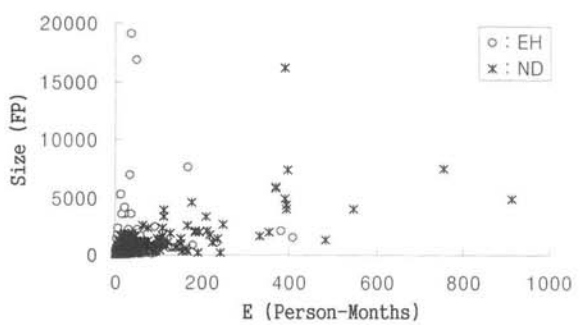
(그림 5)의 기준에 의해 ISBSG Benchmark Release 8[22]로부터 추출된 데이터는 <표 6>과 (그림 6)과 같다.

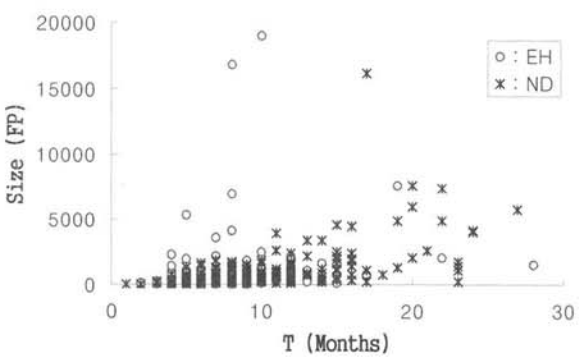


(그림 5) 개발일정 모델 제시를 위한 표본 데이터 추출 범위

〈표 6〉 추출된 표본 현황

개발 구분	Lower Limit	Sampling Range	Upper Limit
성능개량 프로젝트	363건	323건	189건
신규개발 프로젝트	195건	291건	234건





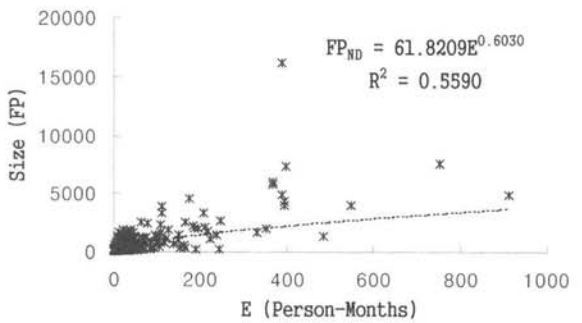
(그림 6) 실험에 적용되는 데이터

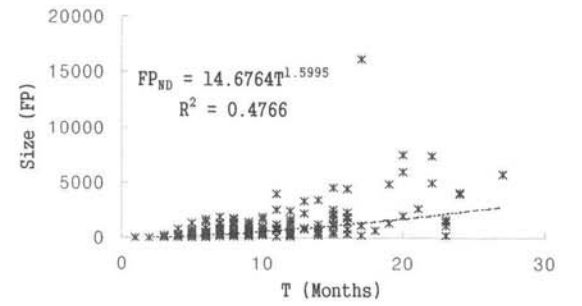
3.2 *E vs. FP*와 *T vs. FP* 관계 유도

추출된 데이터에 기반하여 $FP \propto E^a$와 $FP \propto T^b$ 관계를 회귀분석을 이용하여 유도한 결과, <표 7>의 결과를 얻었다. 예로, 신규개발 프로젝트에 대한 회귀분석 결과는 (그림 7)에 제시하였다.

〈표 7〉 $FP \propto E^a$와 $FP \propto T^b$ 관계

구분		*FP vs. E* 관계	*FP vs. T* 관계
성능개량 프로젝트	모델	$FP = 63.5525E^{0.5292}$	$FP = 19.0300T^{1.3569}$
	상관계수	0.2366(23.66%)	0.2018(20.18%)
	결정계수	0.2764(27.64%)	0.2435(24.35%)
신규개발 프로젝트	모델	$FP = 61.8209E^{0.6030}$	$FP = 14.6764T^{1.5995}$
	상관계수	0.7128(71.28%)	0.5959(59.59%)
	결정계수	0.5590(55.90%)	0.4766(47.66%)





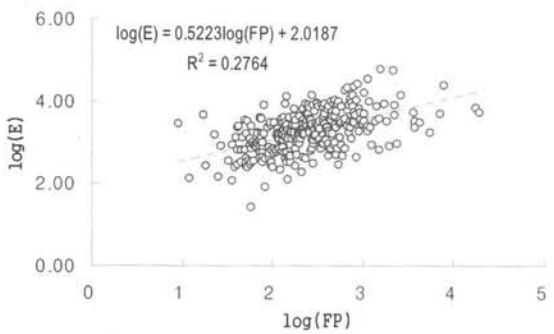
(그림 7) 신규개발 프로젝트의 규모와 노력, 기간 관계

3.3 숙련도 인자 (B) 유도

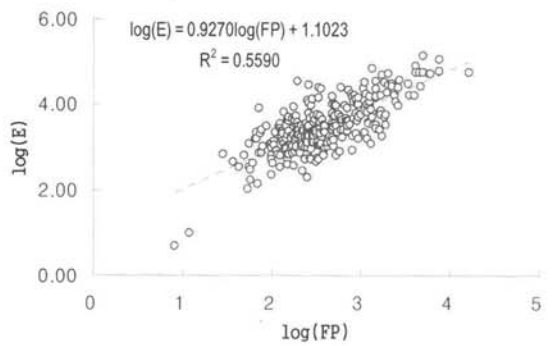
숙련도 인자 B는 소프트웨어 규모에 따라 투입되는 노력과 관련이 있다. 따라서 기능점수에 대한 적절한 B 값을 얻기 위해 (그림 8)의 $log(E)$ *vs.* $log(FP)$ 관계식의 회귀분석을 하였다. 여기서 $log(E)$의 최대 값은 성능개량 프로젝트는 4.38, 신규개발 프로젝트는 5.08이다. 이 값을 적용하여 식 (4)와 같이 B 값을 유도하였다.

$$\text{성능개량 프로젝트} : B = \frac{0.5223 \cdot log(FP) + 2.0187}{4.38 \times 0.4}$$

$$\text{신규개발 프로젝트} : B = \frac{0.9270 \cdot log(FP) + 1.1023}{5.08 \times 0.4} \quad (4)$$



(a) 성능개량 프로젝트



(b) 신규개발 프로젝트

(그림 8) B 값을 유도하기 위한 log(E) vs. log(FP) 관계

3.4 FP 기반 소프트웨어 공식 유도

결론적으로 성능개량 프로젝트는 $FP = E^{0.53} \cdot T^{1.36}$을, 신규개발 프로젝트는 $FP = E^{0.60} \cdot T^{1.60}$의 관계를 갖는다고 할 수 있다. 성능개량 프로젝트는 $0.5 : 1.4 = \frac{1}{2} : \frac{7}{2}$, 신규개발 프로젝트는 $0.6 : 1.6 = \frac{3}{5} : \frac{8}{5}$가 된다. 따라서 기능점수에 기반한 소프트웨어 공식은 식 (5)로 표현된다.

$$\text{성능개량 프로젝트} : FP = PP \cdot (\frac{E}{B})^{\frac{1}{2}} \cdot T^{\frac{7}{2}},$$

$$\text{신규개발 프로젝트} : FP = PP \cdot (\frac{E}{B})^{\frac{3}{5}} \cdot T^{\frac{8}{5}} \quad (5)$$

여기서 E는 월 인원, T는 월을 단위로 사용한다. 비용-일정을 타협하기 위해서는 식 (5)로부터 식 (6)을 유도할 수 있다.

$$\text{성능개량 프로젝트} : E \cdot T^7 = B \cdot (\frac{FP}{PP})^2$$

$$\text{신규개발 프로젝트} : E \cdot T^8 = B \cdot (\frac{FP}{PP})^5 \quad (6)$$

4. 분석 및 평가

4.1 숙련도 인자 (B)

식 (4)로 부터 얻은 B 값의 범위를 소프트웨어 규모(FP)와 관련성을 분석한 결과는 <표 8>과 같다. <표 8>은 <표 1>에 비해 소프트웨어 규모에 대한 B 값을 보다 세밀하게 분류하는 효과를 나타내는 효과를 얻을 수 있다. 따라서, 기존의 *SLOC* 기반의 소프트웨어 공식에서 적용된 B 값의 문제점을 보완할 수 있다.

〈표 8〉 B와 FP 관계

성능개량 프로젝트		신규개발 프로젝트	
FP	B	FP	B
		8	0.15
		-	0.16
		12	0.17
		-	0.18
		28	0.19
		36	0.20
		56	0.21
		76	0.22
9	0.23	106	0.23
18	0.24	146	0.24
27	0.25	200	0.25
49	0.26	273	0.26
78	0.27	378	0.27
127	0.28	514	0.28
206	0.29	699	0.29
338	0.30	939	0.30
548	0.31	1,334	0.31
885	0.32	1,828	0.32
1,387	0.33	2,406	0.33
2,318	0.34	3,403	0.34
3,639	0.35	4,562	0.35
5,291	0.36	6,000	0.36
7,633	0.37	7,555	0.37
-	0.38	-	0.38
19,060	0.39	16,148	0.39
-	0.40	-	0.40

4.2 PP와 PI

Putnam과 Myers[3-6]가 제안한 *SLOC* 기반의 소프트웨어 공식은 개발노력이 년 인원, 개발기간이 년 단위를 사용하고 있다. 반면에 본 논문에서 제안한 PP 기반의 소프트

웨어 공식은 개발노력을 월 인원, 개발기간을 월 단위를 사용한다. 제안된 소프트웨어 공식에 대해 개발노력과 개발기간의 단위를 년/월로 하였을 경우 *PP*의 변화는 <표 9>에 제시하였다.

개발노력을 년 인원으로, 개발기간을 년 단위로 하였을 경우 *PI*의 값 범위를 적절히 표현하지 못하고 있다. 왜냐하면 성능개량 프로젝트는 $\frac{13}{40}=32.5\%$, 신규개발 프로젝트는 $\frac{8}{40}=20\%$ 밖에 표현하지 못하기 때문이다. 그러나 개발노력을 월 인원으로, 개발기간을 월 단위로 하였을 경우 *PP*를 *PI* 값으로 굳이 변화시킬 필요가 없어진다. 즉, *PP* 값을 보면 성능개량 프로젝트는 0.1에서 97.5로 백분위를 표현하고 있으며, 신규개발 프로젝트는 0.02부터 15까지의 범위를 갖기 때문이다. 따라서 *PI*를 사용하지 않고 *PP* 값 자체만으로도 생산성 수준 지표를 표현할 수 있는 장점을 갖고 있다.

〈표 9〉 개발노력과 개발기간의 단위 변경에 따른 *PP* 값 변화

구분	*PP* (*PI*)			
	노력(월 인원), 기간(월) 적용시		노력(년 인원), 기간(년) 적용시	
	최소값	최대값	최소값	최대값
성능개량	0.10 (1)	97.5 (1)	7.22 (1)	10,955 (13)
신규개발	0.02 (1)	15.0 (1)	4.98 (1)	3,541 (8)

4.3 적용 효과

본 제안된 모델을 적용할 경우 얻는 효과는 다음과 같다.

(1) 개발노력을 월 인원, 개발기간을 월 단위를 사용함으로서 *PP*를 *PI* 값으로 변경할 필요가 없다.

(2) 기존 SLOC 기반의 소프트웨어 공식에서 소프트웨어 규모에 따른 *B* 값의 범위를 적절히 표현하지 못하는 단점을 해결할 수 있다.

(3) 기능점수 기반의 소프트웨어 개발시 *FP vs. E*와 *FP vs. T*의 관계를 정립하였다.

(4) 개발언어에 종속되지 않는 소프트웨어 공식 개발로 사용되는 개발언어별 *FP*변환이 불필요하다.

5. 결론 및 향후 연구과제

소프트웨어를 개발시 일반적으로 고객은 보다 빠르게, 보다 저렴하게 개발하도록 압력을 가하고 있다. 그러나 실제적으로 소프트웨어의 개발기간을 더 이상 단축시킬 수 없는 개발 불가능 영역이 존재하며, 개발노력과 일정 간에 특정한 관계를 갖고 있다. 이 개념을 알고 있어야만 소프트웨어의 성공적인 개발이 가능하며, 입찰이나 계역 시 보다 타당한 정보를 가질 수 있다. 이와 같이 개발될 소프트웨어의 규모에 따른 개발노력과 개발기간의 관계를 표현하는 식이 소프트웨어 공식이다. 그러나 기존에 제안된 소프트웨어 공식은 소프트웨어의 규모 단위로 *SLOC*를 사용하고 있다. *SLOC* 자체의 문제점 뿐 아니라 기존의 소프트웨어 공식이 갖고 있는 문제점으로 인해 신규로 개발될 소프트웨어에 대한 적절한 개발노력과 개발기간을 성정하기가 쉽지 않다. 이러한 문제점을 해결하고자, 본 논문에서는 기능점수 규모에 기반한 소프트웨어 공식을 제안하였다.

제안된 소프트웨어 공식은 먼저, 기능점수에 기반한 소프트웨어 공식 유도의 문제점을 제기하고, 이를 해결하기 위해 실험에 적용될 데이터 표본을 추출하는 방법을 제안하였다. 데이터 표본은 실제로 가능한 기간 내에 개발이 종료된 프로젝트를 대상으로 층화 표본추출 방법을 적용하였다. 추출된 표본 데이터를 대상으로 소프트웨어 규모와 개발노력, 규모와 개발기간과의 관계로부터 소프트웨어 공식을 유도하였다. 또한, 기존 소프트웨어 공식이 갖고 있는 문제점인 숙련도 인자를 유도하는 방법을 제안하였다.

제안된 소프트웨어 공식을 적용하면 현재 일반적으로 사용되고 있는 소프트웨어 규모인 기능점수에 적합하며, 개발노력과 개발기간을 월 단위로 활용할 수 있고, *SLOC*를 *FP*로 변환시킬 필요가 없으며, 생산성 수준 지표인 *PI*를 별도로 사용할 필요가 없는 장점을 갖고 있다.

본 논문은 기능점수 규모에 한정된 소프트웨어 공식을 유도하였다. 그러나 소프트웨어 규모 단위로 *FFP*와 *UCP* 등이 점차 확대 적용되고 있는바, 이들 규모 측정 기법을 반영한 소프트웨어 공식들이 추후 요구될 것이다. 따라서 추후 이 분야에 대한 연구를 수행할 예정이다.

참 고 문 헌

[1] A. J. Shenher, "Improving PM: Linking Success Criteria to Project Type," Project Management Institude, Creating Canadian Advantage through Project Management Symposium, Calgory, 1996.

[2] L. H. Putnam and W. Myers, "Familiar Metric Management - Time-to-Market, http://www.qsm.com/fmm_08.pdf

[3] L. H. Putnam and W. Myers, "Five Core Metrics: The Intelligence Behind Successful Software Management," Dorset House Publishing, 2003.

[4] L. H. Putnam and W. Myers, "Measures of Excellence: Reliable Software on Time, Within Budget," Yourdon Press, 1992.

[5] L. H. Putnam and W. Myers, "What We Have Learned," The Journal of Defense Software Engineering, 2000.

[6] L. H. Putnam and W. Myers, "Familier Metric Management +," http://www.qsm.com/ffmm_02.pdf

[7] J. E. Matson, B. E. Barrett and J. M. Mellichamp, "Software

Development Cost Estimation Using Function Points," IEEE Trans. on Software Eng., Vol.20, No.4, pp.275-287, 1994.

[8] C. Jones, "Programming Productivity," New York, McGraw-Hill, 1986.

[9] G. C. Low and D. R. Jeffery, "Function Points in the Estimation and Evaluation of the Software Process," IEEE Trans. on Software Eng., Vol.16, pp.64-71, 1990.

[10] R. D. Emrick, "In Search of a Better Metric for Measuring Productivity of Application Development," Int. Function Point Users Group Conf. Proc., 1987.

[11] B. W. Boehm, "Software Engineering Economics," Prentice Hall, 1981.

[12] M. Bradley, "Function Point Counting Practices Manual, Release 4.1," International Function Point Users Group (IFPUG), 1999.

[13] C. Symons, "COSMIC-FFP Measurement Manual, Version 2.2 (The COSMIC Implementation Guide for ISO/IEC 19761: 2003)," Common Software Measurement International Consortium, 2003.

[14] ISO/IEC FDIS 19761, "Software Engineering - COSMIC-FFP - A Functional Size Measurement Method," 2002.

[15] B. W. Boehm et al, "Software Cost Estimation with COCOMO II," Prentice-Hall, 2000.

[16] K. Ribu, "Estimating Object-oriented Software Projects with Use Cases," University of Oslo Department of Informatics, Master of Science Thesis, 2001.

[17] F. P. Brooks, "The Mythical Man-Month: Essays on Software Engineering," Addison-Wesley, 1995.

[18] J. Nyman, "Metrics and Measures," GlobalTester, TechQA, 2002.

[19] QSM, "Index Based Productivity Benchmarking," QSM Associates, Inc. 1999.

[20] T. McGilbon, "Modern Empirical Cost & Schedule Estimation Tools," DoD DACS Technical Reports, 1997.

[21] D, L, Hallowell, "Software Project Management Meets Six Sigma," http://software.isixsigma.com/library/content/c030813a.asp

[22] ISBSG, "Worldwide Software Development - The Benchmark Release 8," Victoria, Australia International Software Benchmarking Standards Group, 2004.

[23] S. J. Garg, "Software Engineering: Project Management Concepts, Software Metrics," http://www.indyan.com/CS10.html 2003.

[24] QSM, "QSM Function Point Programming Languages Table," Quantitative Software Management, Inc., 2002.

[25] D. Herron, "Software Sizing: There is an Easier Way," David Consulting Group, ASM, 2002.

[26] C. Jones, "Applied Software Measurements," McGraw Hill, 1996.

[27] T. McGilbon, "Modern Empirical Cost & Schedule Estimation Tools," DoD DACS Technical Reports, 1997.

[28] G. Sifri, "Accurate Estimates Critical for Software Development Projects," ESI International, Inc., 2001

[29] W. G. Cochran, "Sampling Techniques," 3rd Edition, John Wiley & Sons, 1977.

[30] D. R. Jones, "Project Scheduling," Augsburg College, 1999.

[31] C. Carbno, "Optimal Resource Allocation for Projects," Project Management Journal, 1999.

[32] Y. Yang, Z. Chen, R. Valerd, and B. Boehm, "Effect of Schedule Compression on Project Effort," 27th Annual Conference of the International Society of Parametric Analysis, Denver, Colorado, USA., 2005.

[33] C. Fakharzadeh, "CORADMO in 2001: A RAD Odyssey," 16th International Forum on COCOMO and Software Cost Modeling, USC-CSE, 2001.

[34] B. Boehm, B. Clark, E. Horowitz, R. Modachy, R. Shelby, and C. Westland, "The COCOMO 2.0 Software Cost Estimation Model," USC Center for Software Engineering, 1995.

[35] J. A. Ward, "Productivity Through Project Management: Controlling the Project Variables," Information Management, 1994.

[36] R. Pressman, C. Jones, V. Basilli, and L. Putnam, "16 Critical Software Practices, Estimate Cost and Schedule Empirically," http://www.iceinusa.com/16CSP/content/2-cost/anmetrgt.html, 2005.

[37] L. K. Richard, "Staff IT," http://www.gantthead.com, 2005.

[38] M. Holtsinger, "CIS 4251/CIS 5930 Software Development," 1999.

[39] C. Abts, A. W. Brown, S. Chulani, B. K. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece, "Software Cost Estimation with COCOMO II," Prentice-Hall, 2000.

[40] Marin Consultancy, "Estimation," Marin Solutions Technical Paper, 2001.

[41] C. Walston and C. Felix, "A Method of Programming Measurement and Estimation," IBM Systems Journal, Vol. 16, No.1, 1977.

[42] K. Johnson, "Software Cost Estimation: Metrics and Models," Department of Computer Science, University of Calgary. 1998.

이 상 운

e-mail : sulee@gwnu.ac.kr

1983년~1987년 한국항공대학교 항공전자공학과(학사)

1995년~1997년 경상대학교 컴퓨터과학과(석사)

1998년~2001년 경상대학교 컴퓨터과학과(박사)

2003년 강원도립대학 컴퓨터응용과 전임강사

2004년~2007년 2월 국립 원주대학 여성교양과 조교수

2007년 3월~현 재 강릉원주대학교 과학기술대학 멀티미디어공학과 부교수

관심분야 : 소프트웨어 프로젝트 관리, 소프트웨어 개발 방법론, 소프트웨어 척도 (소프트웨어 규모, 개발노력, 개발기간, 팀 규모), 분석과 설계 방법론, 소프트웨어 시험 및 품질 보증, 소프트웨어 신뢰성, 신경망, 뉴로-퍼지, 그래프 알고리즘